**ReadMe**

How to execute the code

To execute the code, the user should open a terminal within the folder containing the Python files. In the terminal, the user should type "python3 userinterface.py" and press "ENTER". Doing so will open a text-based user interface via which users can interact with the software by following the menu and pressing a number between 1 and 5. If the user desires to use the navigation system, they have to choose one of the locations below:

* The Pearl
* Landmark Mall
* Doha Festival City
* Mall of Qatar
* Corniche
* Souq Waqif
* City Center
* Doha Exhibition Center
* Aspire

Approach Evaluation and Description of Implementation

As I evaluate my approach, I will occasionally speak in the first person within this document.

First, I created Python files for each of the primary three operations and additional Python files which would host the navigational data the car and its methods would use. Using this structure would improve the reusability and readability of the code by containing each operation and its associated classes within their own files.

Secondly, I created the four main classes (UserInterface, RoutePlanner, MaintenanceMonitor, and EnergyMonitor).

Initially, I expected this task to be relatively easy. As I had already created the UML diagrams for this software, I imagined the coding process to be straightforward and clear. First, I created individual Python files for each of the four main aspects of this software, namely the "UserInterface", "RoutePlanner", "MaintenanceMonitor", and "EnergyMonitor" classes. Theoretically, this would allow me to structure the code more efficiently and improve its readability and reusability. This was the correct move, as I could quickly change the classes and their methods without searching through one large file to find what I was looking for. I first coded the "EnergyMonitor" class, including the "Battery" class, which is closely associated with it. After that, I programmed the "MaintenanceMonitor" and its associated classes, followed by the "RoutePlanner".

For the program's overall functionality and to prove its effectiveness, I included my version of Dijkstra's path-finding algorithm, allowing it to simulate car navigation relatively accurately. To achieve this, I referenced several books and articles mentioned in the reference section below. I created a node network based on the city of Doha. The user can choose from one of the locations within the network; the car will then calculate the shortest path and, if it has enough battery, move to this location. This was the most time-consuming and challenging part of the development process. However, it is incredibly beneficial for the software's overall functionality. Because of my choice to implement Dijkstra's algorithm, I decided against using the "Location" and "Road" classes I had initially designed. They are still present in the code but are not currently utilized.

Finally, I added to my test cases to ensure that every class and method behaved the way I expected, including the text-based user interface discussed above. Not a part of the original design, I later found that my plan needed a proper way for the user to interact with the software. To solve this problem, I coded this specific function. When userinterface.py is executed, this function runs in an infinite loop until the user exits it.

Overall, I am satisfied with my design and implementation. In the future, I want to improve how I test my code. While I did write tests throughout the coding process, I mainly focused on testing after I had already programmed the software. This made testing challenging, and it was sometimes difficult to find out what was causing the error when something went wrong before I had written the tests.

**References for Dijkstra’s Algorithm (as referenced within routeplanner.py)**

Bhargava, A. (2016) *Grokking Algorithms: An Illustrated Guide for Programmers and Other Curious People*. New York: Manning.

GeeksforGeeks (2023) Python Program for Dijkstra’s shortest path algorithm | Greedy Algo-7. Available from: <https://www.geeksforgeeks.org/python-program-for-dijkstras-shortest-path-algorithm-greedy-algo-7/> [Accessed 2 April 2023].

StackAbuse (2023) Dijkstra’s Algorithm. Available from: <https://stackabuse.com/courses/graphs-in-python-theory-and-implementation/lessons/dijkstras-algorithm/> [Accessed 2 April 2023]

AlgoDaily (N.D.) An Illustrated Guide to Dijkstra’s Algorithm. Available from: <https://algodaily.com/lessons/an-illustrated-guide-to-dijkstras-algorithm/python> [Accessed 3 April 2023]